Text Classification:

Data

1. we have total of 20 types of documents(Text files) and total 18828 documents(text files)
2. You can download data from this link, in that you will get documents.rar folder.
If you unzip that, you will get total of 18828 documnets. document name is defined as'Class
so from document name, you can extract the label for that document.
4. Now our problem is to classify all the documents into any one of the class.
5. Below we provided count plot of all the labels in our data.

```
from google.colab import drive                #mounting the google drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
!pip install bs4
!pip install tensorflow_addons
```

```
Requirement already satisfied: bs4 in /usr/local/lib/python3.7/dist-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist-packages
Collecting tensorflow_addons
  Downloading tensorflow_addons-0.14.0-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x8
     |████████████████████████████████| 1.1 MB 12.3 MB/s
Requirement already satisfied: typeguard>=2.7 in /usr/local/lib/python3.7/dist-packages
Installing collected packages: tensorflow-addons
Successfully installed tensorflow-addons-0.14.0
```

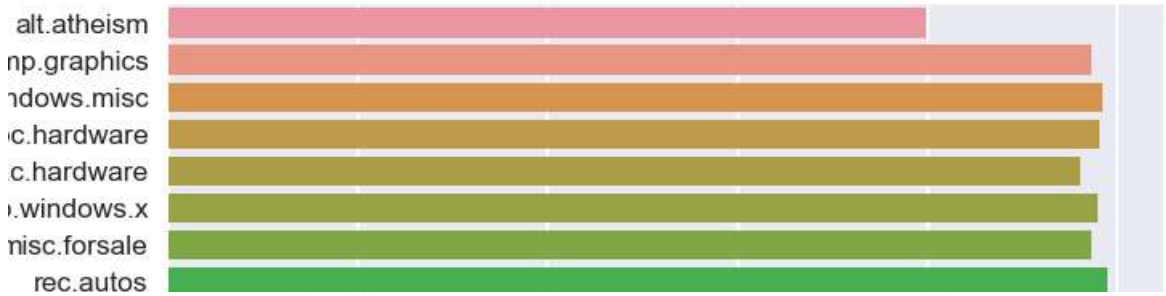
```
import re                                                    #Importing the required lib
import os
import pandas as pd
import nltk
from nltk import ne_chunk, pos_tag, word_tokenize,sent_tokenize
from nltk.tree import Tree
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
from nltk.chunk import tree2conlltags
from sklearn.model_selection import train_test_split
from bs4 import BeautifulSoup
import numpy as np
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input,Dense,Conv1D,Flatten,Embedding,MaxPool1D,concatenat
from tensorflow.keras.callbacks import ModelCheckpoint,TensorBoard,EarlyStopping
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
import pickle
from tensorflow.keras.layers import Conv1D,Flatten,Embedding,MaxPool1D,concatenate,Dropout
from tensorflow.keras.callbacks import ReduceLROnPlateau

import tensorflow_addons as tfa
from tensorflow_addons.metrics import F1Score
from sklearn.metrics import f1_score
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import  ModelCheckpoint
from tensorflow.keras.callbacks import ReduceLROnPlateau
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
```

```
%load_ext tensorboard
```

```
### count plot of all the class labels.
```



Assignment:

sample document

```
Subject: A word of advice
From: jcopelan@nyx.cs.du.edu (The One and Only)

In article < 65882@mimsy.umd.edu > mangoe@cs.umd.edu (Charley Wingate) writes:
>
>I've said 100 times that there is no "alternative" that should think you
>might have caught on by now.  And there is no "alternative", but the point
>is, "rationality" isn't an alternative either.  The problems of metaphysical
>and religious knowledge are unsolvable-- or I should say, humans cannot
>solve them.

How does that saying go: Those who say it can't be done shouldn't interrupt
those who are doing it.

Jim
--
Have you washed your brain today?
```

Preprocessing:

useful links: http://www.pyregex.com/

1. Find all emails in the document and then get the text after the "@". and then split thos
after that remove the words whose length is less than or equal to 2 and also remove'com' wo

In one doc, if we have 2 or more mails, get all.
Eg:[test@dm1.d.com, test2@dm2.dm3.com]-->[dm1.d.com, dm3.dm4.com]-->[dm1,d,com,dm2,dm3,com]
append all those into one list/array. (This will give length of 18828 sentences i.e one li
Some sample output was shown below.

> In the above sample document there are emails [jcopelan@nyx.cs.du.edu, 65882@mimsy.umd.ed

preprocessing:
[jcopelan@nyx.cs.du.edu, 65882@mimsy.umd.edu, mangoe@cs.umd.edu] ==> [nyx cs du edu mimsy u
[nyx edu mimsy umd edu umd edu]

2. Replace all the emails by space in the original text.

```
# we have collected all emails and preprocessed them, this is sample output
preprocessed_email
```

```
array(['juliet caltech edu',
       'coding bchs edu newsgate sps mot austlcm sps mot austlcm sps mot com  dna bchs e
       'batman bmd trw', ..., 'rbdc wsnc org dscomsa desy zeus  desy',
       'rbdc wsnc org morrow stanford edu pangea Stanford EDU',
       'rbdc wsnc org apollo apollo'], dtype=object)
```

```
len(preprocessed_email)
```

```
18828
```

3. Get subject of the text i.e. get the total lines where "Subject:" occur and remove
the word which are before the ":" remove the newlines, tabs, punctuations, any special char
Eg: if we have sentance like "Subject: Re: Gospel Dating @ \r\r\n" --> You have to get "Gos
Save all this data into another list/array.

4. After you store it in the list, Replace those sentances in original text by space.

5. Delete all the sentances where sentence starts with **"Write to:"** or **"From:"**.
> In the above sample document check the 2nd line, we should remove that

6. Delete all the tags like "< anyword >"
> In the above sample document check the 4nd line, we should remove that "< 65882@mimsy.umd

7. Delete all the data which are present in the brackets.
In many text data, we observed that, they maintained the explanation of sentence

or translation of sentence to another language in brackets so remove all those.
Eg: "AAIC-The course that gets you HIRED(AAIC - Der Kurs, der Sie anstellt)" --> "AAIC-The

> In the above sample document check the 4nd line, we should remove that "(Charley Wingate)

8. Remove all the newlines('\n'), tabs('\t'), "-", "\".

9. Remove all the words which ends with ":".
Eg: "Anyword:"
> In the above sample document check the 4nd line, we should remove that "writes:"

10. Decontractions, replace words like below to full words.
please check the donors choose preprocessing for this
Eg: can't -> can not, 's -> is, i've -> i have, i'm -> i am, you're -> you are, i'll --> i

There is no order to do point 6 to 10. but you have to get final output correctly

11. Do chunking on the text you have after above preprocessing.
Text chunking, also referred to as shallow parsing, is a task that
follows Part-Of-Speech Tagging and that adds more structure to the sentence.
So it combines the some phrases, named entities into single word.
So after that combine all those phrases/named entities by separating "_".
And remove the phrases/named entities if that is a "Person".
You can use **nltk.ne_chunk** to get these.
Below we have given one example. please go through it.

useful links:
https://www.nltk.org/book/ch07.html
https://stackoverflow.com/a/31837224/4084039
http://www.nltk.org/howto/tree.html
https://stackoverflow.com/a/44294377/4084039

```
#i am living in the New York
print("i am living in the New York -->", list(chunks))
print(" ")
print("-"*50)
print(" ")
#My name is Srikanth Varma
print("My name is Srikanth Varma -->", list(chunks1))
```

```
i am living in the New York --> [('i', 'NN'), ('am', 'VBP'), ('living', 'VBG'), ('in',
```

```
--------------------------------------------------

My name is Srikanth Varma --> [('My', 'PRP$'), ('name', 'NN'), ('is', 'VBZ'), Tree('PERS
```

We did chunking for above two lines and then We got one list where each word is mapped to a POS(parts of speech) and also if you see "New York" and "Srikanth Varma",
they got combined and represented as a tree and "New York" was referred as "GPE" and "Srika
so now you have to Combine the "New York" with "_" i.e "New_York"
and remove the "Srikanth Varma" from the above sentence because it is a person.

13. Replace all the digits with space i.e delete all the digits.
> In the above sample document, the 6th line have digit 100, so we have to remove that.

14. After doing above points, we observed there might be few word's like
"_word_" (i.e starting and ending with the _), "_word" (i.e starting with the _),
"word_" (i.e ending with the _) remove the _ from these type of words.

15. We also observed some words like **"OneLetter_word"- eg: d_berlin,**
"TwoLetters_word" - eg: dr_berlin , in these words we remove the "OneLetter_" (d_berlin ==> "TwoLetters_" (de_berlin ==> berlin). i.e remove the words
which are length less than or equal to 2 after spliiting those words by "_".

16. Convert all the words into lower case and lowe case
and remove the words which are greater than or equal to 15 or less than or equal to 2.

17. replace all the words except "A-Za-z_" with space.

18. Now You got Preprocessed Text, email, subject. create a dataframe with those.
Below are the columns of the df.

```
data.columns
```

```
Index(['text', 'class', 'preprocessed_text', 'preprocessed_subject',
       'preprocessed_emails'],
      dtype='object')
```

```
data.iloc[400]
```

```
text                    From: arc1@ukc.ac.uk (Tony Curtis)\r\r\r\nSubj...
```

```
class                                                    alt.atheism
preprocessed_text       said re is article if followed the quoting rig...
preprocessed_subject                                christian morality is
preprocessed_emails                                   ukc mac macalstr edu
Name: 567, dtype: object
```

```
#Extracting all documents in drive
os.chdir("/content/drive/MyDrive/Assignment19_Document_Classification_CNN/")
print(os.listdir())
```

```
[]
```

```
!unrar e documents.rar /Documents/
```

```
Streaming output truncated to the last 5000 lines.
Extracting  /Documents/sci.space_60893.txt                            OK
Extracting  /Documents/sci.space_60894.txt                            OK
Extracting  /Documents/sci.space_60895.txt                            OK
Extracting  /Documents/sci.space_60896.txt                            OK
Extracting  /Documents/sci.space_60897.txt                            OK
Extracting  /Documents/sci.space_60898.txt                            OK
Extracting  /Documents/sci.space_60899.txt                            OK
Extracting  /Documents/sci.space_60900.txt                            OK
Extracting  /Documents/sci.space_60901.txt                            OK
Extracting  /Documents/sci.space_60902.txt                            OK
Extracting  /Documents/sci.space_60903.txt                            OK
Extracting  /Documents/sci.space_60904.txt                            OK
Extracting  /Documents/sci.space_60905.txt                            OK
Extracting  /Documents/sci.space_60906.txt                            OK
Extracting  /Documents/sci.space_60907.txt                            OK
Extracting  /Documents/sci.space_60908.txt                            OK
Extracting  /Documents/sci.space_60909.txt                            OK
Extracting  /Documents/sci.space_60910.txt                            OK
Extracting  /Documents/sci.space_60911.txt                            OK
Extracting  /Documents/sci.space_60912.txt                            OK
Extracting  /Documents/sci.space_60913.txt                            OK
Extracting  /Documents/sci.space_60914.txt                            OK
Extracting  /Documents/sci.space_60915.txt                            OK
Extracting  /Documents/sci.space_60916.txt                            OK
Extracting  /Documents/sci.space_60917.txt                            OK
Extracting  /Documents/sci.space_60918.txt                            OK
Extracting  /Documents/sci.space_60919.txt                            OK
Extracting  /Documents/sci.space_60920.txt                            OK
Extracting  /Documents/sci.space_60921.txt                            OK
Extracting  /Documents/sci.space_60922.txt                            OK
Extracting  /Documents/sci.space_60923.txt                            OK
Extracting  /Documents/sci.space_60924.txt                            OK
Extracting  /Documents/sci.space_60925.txt                            OK
Extracting  /Documents/sci.space_60926.txt                            OK
Extracting  /Documents/sci.space_60927.txt                            OK
Extracting  /Documents/sci.space_60928.txt                            OK
Extracting  /Documents/sci.space_60929.txt                            OK
Extracting  /Documents/sci.space_60930.txt                            OK
Extracting  /Documents/sci.space_60931.txt                            OK
Extracting  /Documents/sci.space_60932.txt                            OK
```

```
Extracting  /Documents/sci.space_60933.txt                                OK
Extracting  /Documents/sci.space_60934.txt                                OK
Extracting  /Documents/sci.space_60935.txt                                OK
Extracting  /Documents/sci.space_60936.txt                                OK
Extracting  /Documents/sci.space_60937.txt                                OK
Extracting  /Documents/sci.space_60938.txt                                OK
Extracting  /Documents/sci.space_60939.txt                                OK
Extracting  /Documents/sci.space_60940.txt                                OK
Extracting  /Documents/sci.space_60941.txt                                OK
Extracting  /Documents/sci.space_60942.txt                                OK
Extracting  /Documents/sci.space_60943.txt                                OK
Extracting  /Documents/sci.space_60944.txt                                OK
Extracting  /Documents/sci.space_60945.txt                                OK
Extracting  /Documents/sci.space_60946.txt                                OK
Extracting  /Documents/sci.space_60947.txt                                OK
Extracting  /Documents/sci.space_60948.txt                                OK
Extracting  /Documents/sci.space_60949.txt                                OK
Extracting  /Documents/sci.space_60950.txt                                OK
```

To get above mentioned data frame --> Try to Write Total Preprocessing steps in One Function Named Preprocess as below.

```
def preprocess(file):                                                #preprocess function to do pr
    """Do all the Preprocessing as shown above and
    return a tuple contain preprocess_email,preprocess_subject,preprocess_text for that Text_

    dir="/Documents/"
    doc_class_label = file.split('_')[0]

    with open(dir+file,'rb') as f1:
        preprocessed_emails=[]
        preprocessed_subjects=[]
        text = f1.read()
        org_text = str(text)

        soup = BeautifulSoup(text,'lxml')                                          #Getting cl
        text = soup.get_text()
        match=re.findall(r"[a-z0-9\.\-+_]+@[a-z0-9\.\-+_]+\.[a-z]+",org_text)          #rege
                                                          #Took the below reference to see h

        for mail in  match:                 #loop to do  basic email preprocessing like
                mail=mail.split('@')[1] #getting text after the "@". and then split those tex
                mail=mail.split('.')
                for val in mail:
                      if len(val)>2 and val not in 'com':  #after that remove the words whose
                          preprocessed_emails.append(val)                    #appending the
        finall_mail_st=""
        for mail in preprocessed_emails:
              finall_mail_st+="  " +mail
        if (re.findall(r'Subject:.*',text)):
              subject = re.findall(r'Subject:.*',text)[0].split(":")[-1]    ##searching for a
```

```
      subject = re.sub('[^A-Za-z0-9]+',' ',subject)                 #searching for al
      subject =subject.lower()
else:
    subject= " "                                                     #get the total l
                                                                   #the word which are

text=re.sub(r"[a-z0-9\.\-+_]+@[a-z0-9\.\-+_]+\.[a-z]+"," ",text )
text = re.sub(r'Subject:.*'," ",text)                          #Getting Document text

text= re.sub(r'Write to:.+', '',text)                    #Delete all the sentances whe
text= re.sub(r'From:.+', '',text)

text=re.sub(r'\([^()]*\)', '', text)                     # Delete all the data(words s
text=re.sub(r'\(',"",text)
text=re.sub(r'\)',"",text)
text=re.sub(r'\t',"",text)                      #Removing  all the newlines('\n'), tabs
text=re.sub(r'\-',"",text)
text=re.sub(r'\\',"",text)
text=re.sub(r'\\n',"",text)
text = re.sub(r"\s+"," ",text)
text = re.sub(r"[/]",".",text)
text = re.sub(r"\.","",text)
text = re.sub(r"\+","",text)
text = re.sub(r"\*","",text)
text=re.sub(r'\w+:\s?',"",text)                        #Removing all the words which end
#Implemeting Decontraction of words in preprocessed text
text= re.sub(r"won't", "will not",text)
text = re.sub(r"can\'t", "can not",text)
text= re.sub(r"n\'t", " not",text)
text = re.sub(r"\'re", " are",text)
text= re.sub(r"\'s", " is",text)
text = re.sub(r"\'d", " would",text)
text= re.sub(r"\'ll", " will",text)
text= re.sub(r"\'t", " not",text)
text= re.sub(r"\'ve", " have",text)
text = re.sub(r"\'m", " am",text)

 #Implementing Parts of Peech Tagging to invidual words and then doing Chunking using
chunks=ne_chunk(pos_tag(word_tokenize(text)))
Person_labels=[]
Non_Person_labels=[]
Chunked_text1=ne_chunk(pos_tag(word_tokenize(text)))
Person_labels=[]
Non_Person_labels=[]
for c in Chunked_text1:
    if isinstance(c, Tree) and c.label() == 'PERSON':             #Checking if the ch
        #  The object is <class 'nltk.tree.Tree'> and label is  PERSON  Entity
        Person_labels.append(' '.join([w for w, _ in c.leaves()]))
    if isinstance(c, Tree) and c.label()!='PERSON':                    #Checking if
          if len(c)>2:
               Non_Person_labels.append('_'.join([w for w, _ in c.leaves()]))
```

```
                else:
                    Non_Person_labels.append(c.leaves()[0][0])
        for word in Person_labels:                              #Replacing all the PERSON(Nam
            text=re.sub(word,"",text)
        for word in Non_Person_labels:                          #Joining  all the  NON-PERSON
            old_word_list=word.split("_")
            org_non_person_word=" ".join(old_word_list)
            text=re.sub(org_non_person_word,word,text)


        text=re.sub(r'[0-9]+'," ",text)                #Replacing all the digits with space
        text=re.sub(r'\.',"",text)
        text = re.sub(r"(_?)([A-Za-z0-9])(_?)",r'\2',text) #remove the _ from these type of w
        text = re.sub(r"([A-Za-z]{1,2})(_)(A-Za-z)","\g<3>",text)  #remove the words which ar
        text=re.sub(r"[^A-Za-z_]"," ",text)           #Finally replacing  all the words excep
        text=text.lower()           #Converting all  the words to lower case in preprocessed

        final_preprocessed_text = ' '.join([wrd for wrd in text.split() if len(wrd)>2 and len


        return [org_text,doc_class_label,final_preprocessed_text,subject,finall_mail_st]
```

```
#Running the preprocess function for sample   alt.atheism_49960.txt file
dir="/Documents/"  #Directory path where all documents are present
sample_dat_lst=preprocess('alt.atheism_49960.txt')
sample_dat_lst
```

```
['b\'From: mathew <mathew@mantis.co.uk>\\nSubject: Alt.Atheism FAQ: Atheist Resources\\r
 'alt.atheism',
 'resources december atheist organizations usa freedom from religion foundation fish bum
 ' atheist resources',
 '  mantis  netcom  mantis']
```

```
#Preprocessing every document.
dir="/Documents/"      #directory where all file are present in drive
f_processed=0                    #counter variable
data_list=[]                     #finla list to store data of each document as list
for file in os.listdir(dir):                          #loop to iterate  over each file
      if f_processed%1000==0:
         print("Number of files processed:",f_processed)
      if f_processed==18828:
          print("All files processed",f_processed)
```

```
        f_processed+=1
        final_dat=preprocess(file)                #calling the text preprocess() function
        data_list.append(final_dat)               #appending the data to final list for each fi
```

```
      Number of files processed: 0
      Number of files processed: 1000
      Number of files processed: 2000
      Number of files processed: 3000
      Number of files processed: 4000
      Number of files processed: 5000
      Number of files processed: 6000
      Number of files processed: 7000
      Number of files processed: 8000
      Number of files processed: 9000
      Number of files processed: 10000
      Number of files processed: 11000
      Number of files processed: 12000
      Number of files processed: 13000
      Number of files processed: 14000
      Number of files processed: 15000
      Number of files processed: 16000
      Number of files processed: 17000
      Number of files processed: 18000
```

```
#Making the final dataframe
final_df1 = pd.DataFrame(data_list,columns =['text', 'class', 'preprocessed_text', 'preproces
       'preprocessed_mail'])
```

```
final_df1.shape     #Checking the shape of final data
```

```
     (18828, 5)
```

```
final_df1.head()                  #Reading the df data
```

	text	class	preprocessed_text	preprocessed_
0	b"From: ricardo@rchland.vnet.ibm.com (Ricardo ...	comp.graphics	article quvdoinn concerning the proposed newsg...	newsg
1	b"From: holmes7000@iscsvax.uni.edu\nSubject: A...	comp.graphics	what the best way archive gif zip them and the...	arc
2	b'From: kxgst1+@pitt.edu (Kenneth Gilbert)\nSu...	sci.med	article chronic persistent hepatitis usually d...	persistent v
	b'From: jadfiche@silver.ucs.indiana.edu		article doing	

```
#Saving the  final dataframe as pickle file.
```

```
os.chdir("/content/drive/MyDrive/Assignment19_Document_Classification_CNN/")
final_df1.to_pickle("final_preprocessed_text_df")
```

```
os.chdir("/content/drive/MyDrive/Assignment19_Document_Classification_CNN/")
final_df2=pd.read_pickle("final_preprocessed_text_df")          #Reading the final dataframe
final_df2.shape
```

```
(18828, 5)
```

Code checking:

After Writing preprocess function. call that functoin with the input text of 'alt.atheism_49960' doc and print the output of the preprocess function
This will help us to evaluate faster, based on the output we can suggest you if there are any changes.

After writing Preprocess function, call the function for each of the document(18828 docs) and then create a dataframe as mentioned above.

Training The models to Classify:

1. Combine "preprocessed_text", "preprocessed_subject", "preprocessed_emails" into one colu

2. Now Split the data into Train and test. use 25% for test also do a stratify split.

3. Analyze your text data and pad the sequnce if required.
Sequnce length is not restricted, you can use anything of your choice.
you need to give the reasoning

4. Do Tokenizer i.e convert text into numbers. please be careful while doing it.
if you are using tf.keras "Tokenizer" API, it removes the "_", but we need that.

5. code the model's (Model-1, Model-2) as discussed below
and try to optimize that models.

6. For every model use predefined Glove vectors.
Don't train any word vectors while Training the model.

7. Use "categorical_crossentropy" as Loss.

8. Use **Accuracy and Micro Avgeraged F1 score** as your as Key metrics to evaluate your model.

9. Use Tensorboard to plot the loss and Metrics based on the epoches.

10. Please save your best model weights in to **'best_model_L.h5' (L = 1 or 2).**

11. You are free to choose any Activation function, learning rate, optimizer.
But have to use the same architecture which we are giving below.

12. You can add some layer to our architecture but you **deletion** of layer is not acceptable.

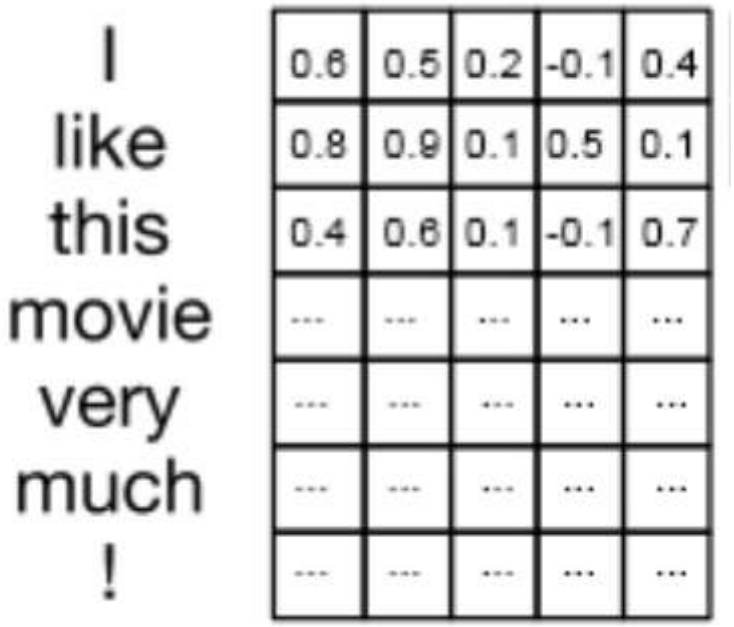
13. Try to use **Early Stopping** technique or any of the callback techniques that you did in t

14. For Every model save your model to image (Plot the model) with shapes
and inlcude those images in the notebook markdown cell,
upload those imgages to Classroom. You can use "plot_model"
please refer this if you don't know how to plot the model with shapes.

Model-1: Using 1D convolutions with word embeddings

Encoding of the Text --> For a given text data create a Matrix with Embedding layer as sho
In the example we have considered d = 5, but in this assignment we will get d = dimension o
i.e if we have maximum of 350 words in a sentence and embedding of 300 dim word vector,

we result in 350*300 dimensional matrix for each sentance as output after embedding layer



Ref: https://i.imgur.com/kiVQuk1.png

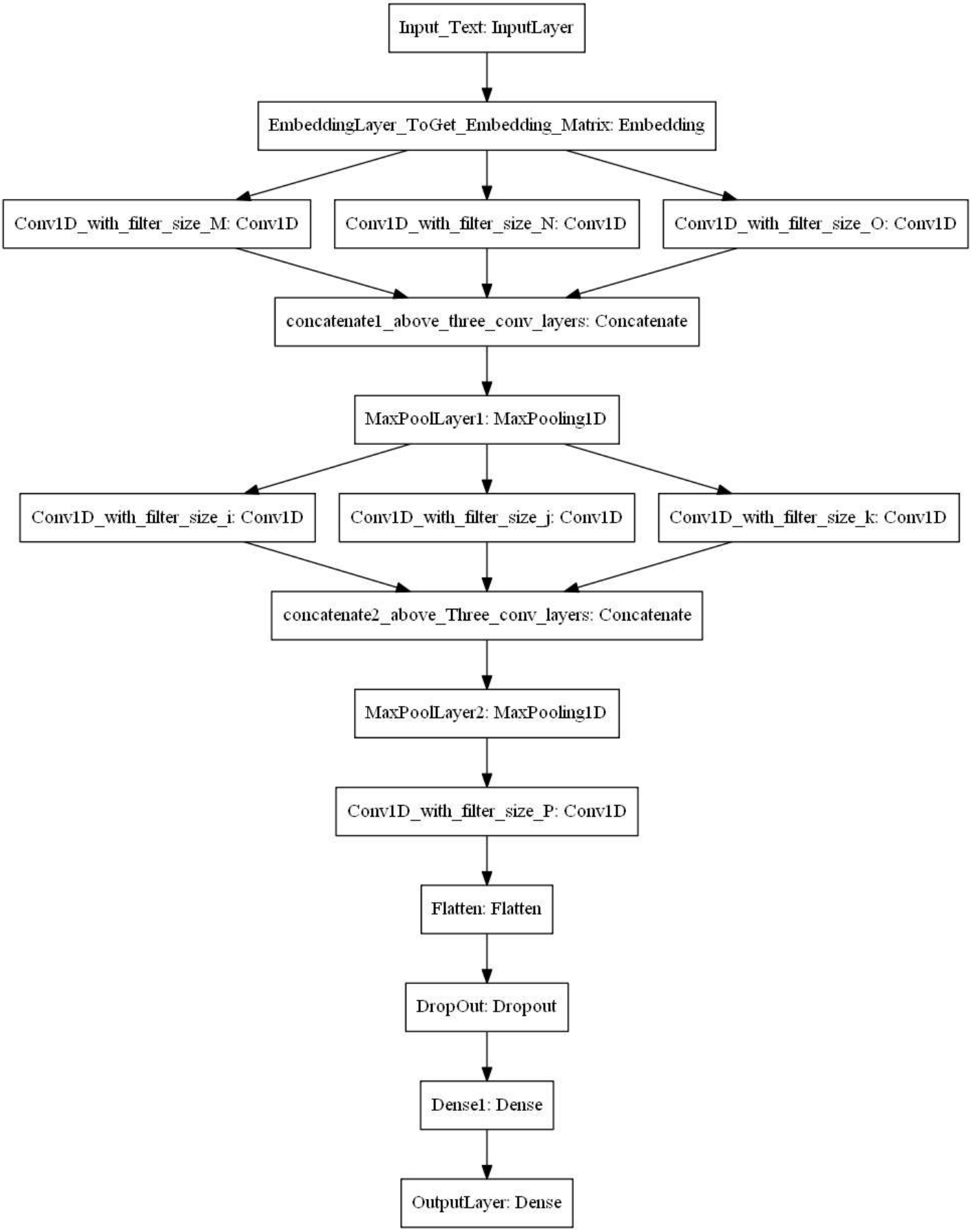
Reference:

https://stackoverflow.com/a/43399308/4084039

https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-k

How EMBEDDING LAYER WORKS

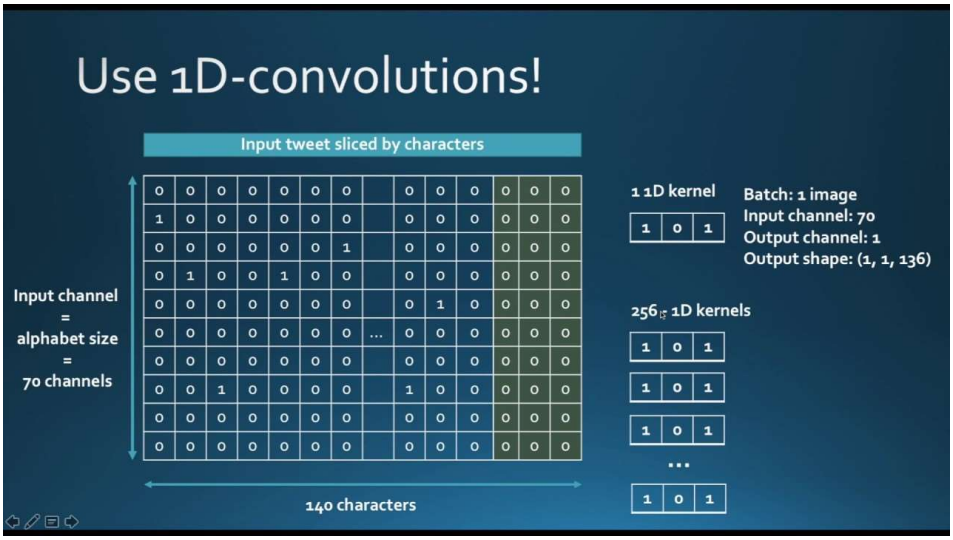
- Go through this blog, if you have any doubt on using predefined Embedding values in Embedding layer - https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/



ref: 'https://i.imgur.com/fv1GvFJ.png'

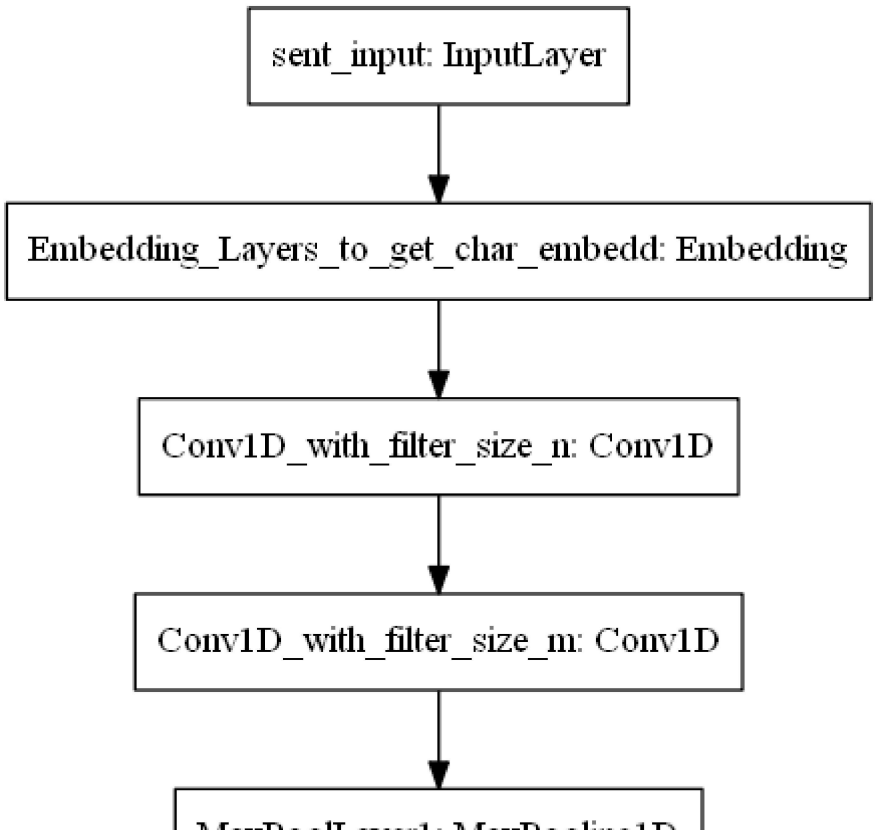
```
1. all are Conv1D layers with any number of filter and filter sizes, there is no restrictio

2. use concatenate layer is to concatenate all the filters/channels.

3. You can use any pool size and stride for maxpooling layer.

4. Don't use more than 16 filters in one Conv layer becuase it will increase the no of para
( Only recommendation if you have less computing power )

5. You can use any number of layers after the Flatten Layer.
```

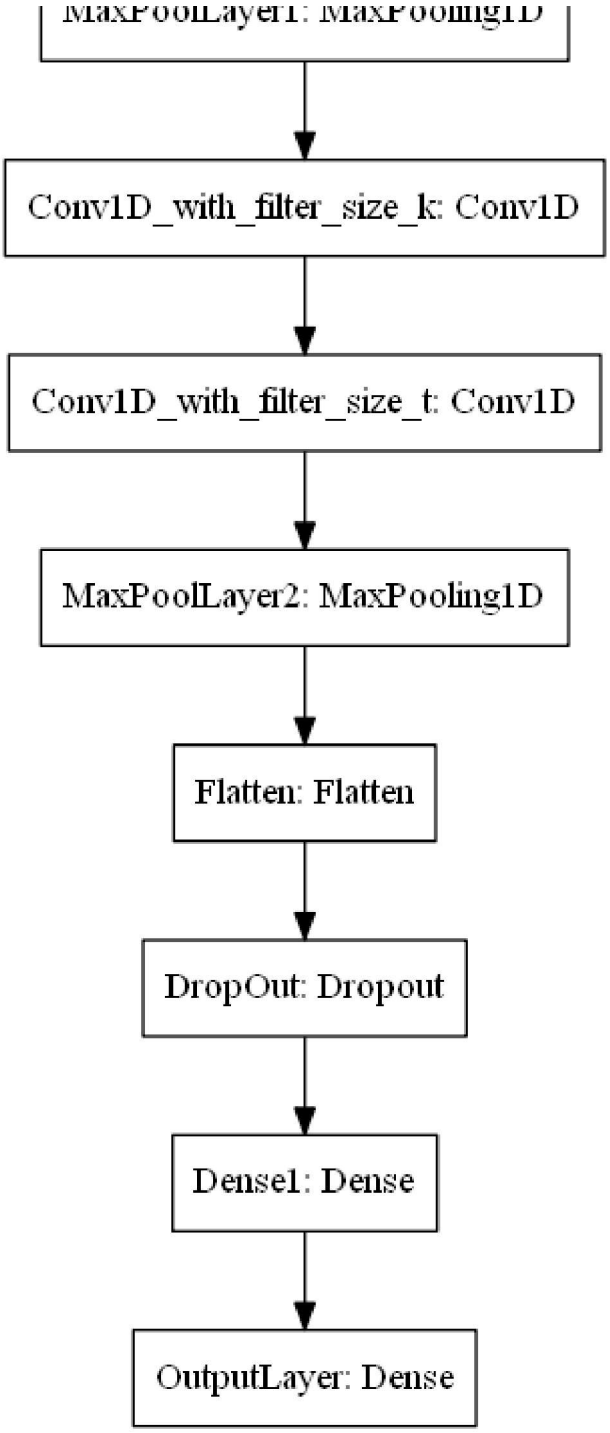
Model-2 : Using 1D convolutions with character embedding



Here are the some papers based on Char-CNN

1. Xiang Zhang, Junbo Zhao, Yann LeCun. Character-level Convolutional Networks for Text
2. Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush. Character-Aware Neural Lar
3. Shaojie Bai, J. Zico Kolter, Vladlen Koltun. An Empirical Evaluation of Generic Convc
4. Use the pratrained char embeddings https://github.com/minimaxir/char-embeddings/blob/






```
#Combining preprocessed text  ,subject and email and  creating df column for same
final_df2['pre_text_pre_suject_pre_email']=final_df2['preprocessed_text']+final_df2['preproce
final_df2.head()
```

	text	class	preprocessed_text	preprocessed_
0	b"From: ricardo@rchland.vnet.ibm.com (Ricardo ...	comp.graphics	article quvdoinn concerning the proposed newsg...	newsg
1	b"From: holmes7000@iscsvax.uni.edu\nSubject: A...	comp.graphics	what the best way archive gif zip them and the...	arc

```
#checking the datatype  of df columns
final_df2.dtypes
```

```
text                           object
class                          object
preprocessed_text              object
preprocessed_subject           object
preprocessed_mail              object
pre_text_pre_suject_pre_email    object
dtype: object
```

```
X=final_df2['pre_text_pre_suject_pre_email']
Y=final_df2['class']
#encoding Target Y class labels
Label_en=LabelEncoder()
Label_en.fit(Y)
Label_en_Y=Label_en.transform(Y)
#Converting final labels to  matrix
Y=np_utils.to_categorical(Label_en_Y)
Y.shape
```

```
(18828, 20)
```

```
#Splitting the data into train and test
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25,stratify=Y)
```

```
#Finding the  optimal sequence length
sents_len=[len(sen) for sen in X_train]
sents_len=sorted(sents_len)
sents_len=np.array(sents_len)
```

```
#checking the stats for sequence selection
Median=int(np.percentile(sents_len,95))
Median
```

```
3112
```

```
#Checking 95th percenttile
per_tile_95=int(np.percentile(sents_len,95))
```

```
per_tile_95
```

```
3112
```

```
#Checking 98th percenttile
per_tile_98=int(np.percentile(sents_len,98))
per_tile_98
```

```
5346
```

```
max_length_of_sent=per_tile_98
max_length_of_sent
```

```
5346
```

Since 98 Percent of text sentences are less than 5241 .so we will choose max len=5241

```
ke_tokenizer = Tokenizer(filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n')   # Referred the foll

ke_tokenizer.fit_on_texts(X_train)
X_train  = ke_tokenizer.texts_to_sequences(X_train)
X_test = ke_tokenizer.texts_to_sequences(X_test)
```

```
#Appliyng paddding based on maxlen
X_train=pad_sequences(X_train,max_length_of_sent,padding='post')
X_test=pad_sequences(X_test,max_length_of_sent,padding='post')
```

```
print(X_train.shape)
print(X_test.shape)
```

```
(14121, 5346)
(4707, 5346)
```

```
Y.shape
```

```
(18828, 20)
```

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
```

```
--2021-11-01 05:53:27--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
```

```
--2021-11-01 05:53:27--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2021-11-01 05:53:28--  http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:80...
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip’

glove.6B.zip        100%[===================>] 822.24M  4.98MB/s    in 2m 41s

2021-11-01 05:56:08 (5.12 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]
```

```
print(os.getcwd())
```

```
/content/drive/My Drive/Assignment19_Document_Classification_CNN
```

```
!unzip glove*.zip
```

```
Archive:  glove.6B.zip
  inflating: glove.6B.50d.txt
  inflating: glove.6B.100d.txt
  inflating: glove.6B.200d.txt
  inflating: glove.6B.300d.txt
```

```
#Indexing the word vectors                        #used the below reference to see how load g

embd_dict = {}
f = open('glove.6B.50d.txt', encoding='utf-8')
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embd_dict[word] = coefs
f.close()
```

```
print("No of word vetors ",len(embd_dict))
```

```
No of word vetors  400000
```

```
#Creating embedding Matrix
emd_mat_size = len(ke_tokenizer.word_index)+1
embd_matrix = np.zeros((emd_mat_size,50))

for word,i in ke_tokenizer.word_index.items():
```

```
    emb_word = embd_dict.get(word)
    if emb_word is not None:
        embd_matrix[i]=emb_word



print("Embdedding Matrix shape:",embd_matrix.shape)
```

```
 Embdedding Matrix shape: (83617, 50)
```

1st Model

```
#Defining Embedding Layer
Embd_ly=Embedding(len(ke_tokenizer.word_index)+1,50,embeddings_initializer=tf.keras.initializ


input_layer=Input(shape=(max_length_of_sent,))
embd_layer=Embd_ly(input_layer)
M1=Conv1D(64,4,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),kernel_
N1=Conv1D(64,4,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),kernel_
O1=Conv1D(64,4,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),kernel_
sec_layer=concatenate([M1,N1,O1])
max_pool1=MaxPool1D(3)(sec_layer)
I1=Conv1D(32,8,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),kernel_
J1=Conv1D(32,8,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),kernel_
K1=Conv1D(32,8,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),kernel_
third_layer=concatenate([I1,J1,K1])
max_pool2=MaxPool1D(3)(third_layer)
fourth_layer=Conv1D(32,4,activation='relu',kernel_initializer=tf.keras.initializers.he_normal
flat_layer1=Flatten()(fourth_layer)
dropout_layer1=Dropout(0.2)(flat_layer1)
dense_layer1=Dense(64,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),
Output_layer1=Dense(20,activation='softmax',kernel_initializer=tf.keras.initializers.glorot_n
M1=Model(inputs=input_layer,outputs=Output_layer1)


#Custom callback for F1_score
class f1(tf.keras.callbacks.Callback):
        def __init__(self,validation_data):
              super().__init__()
              self.validation_data=validation_data            #initializing the X_test,Y_test
        def on_train_begin(self,logs={}):
              self.f1_scores=[]                               #list to store f1_scores

        def on_epoch_end(self,epoch,logs={}):
            y_predicted =self.model.predict(self.validation_data[0]).argmax(axis=1)   #gettin
            y_true=np.zeros(Y_test.shape[0])
            for x in range(len(y_true)):
                y_true[x]=int(np.argmax(Y_test[x]))
```

```
            val_f1=f1_score(y_true, y_predicted,average='micro')          #computin f1_score

            self.f1_scores.append(val_f1)
            print(" Validation F1 score {}".format(val_f1))

M1.summary()
```

```
Model: "model_2"
```

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	[(None, 5346)]	0	
embedding_1 (Embedding)	(None, 5346, 50)	4180850	input_3[0][0]
conv1d_14 (Conv1D)	(None, 5343, 64)	12864	embedding_1[2][0]
conv1d_15 (Conv1D)	(None, 5343, 64)	12864	embedding_1[2][0]
conv1d_16 (Conv1D)	(None, 5343, 64)	12864	embedding_1[2][0]
concatenate_4 (Concatenate)	(None, 5343, 192)	0	conv1d_14[0][0] conv1d_15[0][0] conv1d_16[0][0]
max_pooling1d_4 (MaxPooling1D)	(None, 1781, 192)	0	concatenate_4[0][0]
conv1d_17 (Conv1D)	(None, 1774, 32)	49184	max_pooling1d_4[0][0]
conv1d_18 (Conv1D)	(None, 1774, 32)	49184	max_pooling1d_4[0][0]
conv1d_19 (Conv1D)	(None, 1774, 32)	49184	max_pooling1d_4[0][0]
concatenate_5 (Concatenate)	(None, 1774, 96)	0	conv1d_17[0][0] conv1d_18[0][0] conv1d_19[0][0]
max_pooling1d_5 (MaxPooling1D)	(None, 591, 96)	0	concatenate_5[0][0]
conv1d_20 (Conv1D)	(None, 588, 32)	12320	max_pooling1d_5[0][0]
flatten_2 (Flatten)	(None, 18816)	0	conv1d_20[0][0]
dropout_2 (Dropout)	(None, 18816)	0	flatten_2[0][0]
dense_4 (Dense)	(None, 64)	1204288	dropout_2[0][0]
dense_5 (Dense)	(None, 20)	1300	dense_4[0][0]

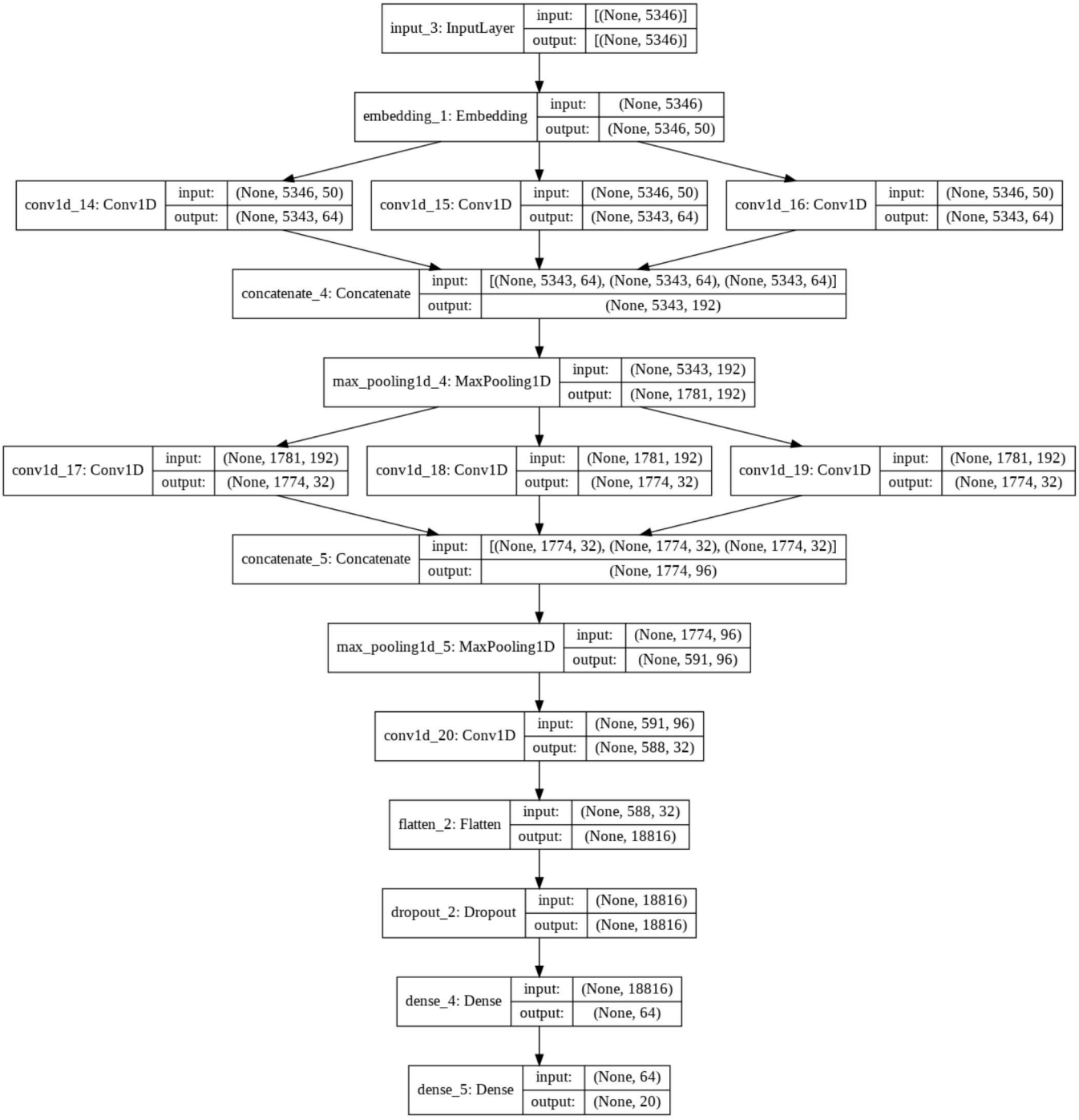
```
Total params: 5,584,902
Trainable params: 1,404,052
Non-trainable params: 4,180,850
```

```
## f1_score_callback
f1_metric=f1((X_test,Y_test))

## Callback for saving best model
checkpoint = ModelCheckpoint(filepath='best_model_1.h5',verbose=1,monitor='val_accuracy',
                             mode='max',save_best_only=True)

## Callback for earlystopping
early_stop = EarlyStopping(monitor="val_accuracy",mode='max',patience=2)

## Tensorboard
log_dir = "logs"
tensorboard_cb = TensorBoard(log_dir=log_dir,histogram_freq=1,write_graph=True)


## all callbacks
callbacks =[f1_metric,early_stop,tensorboard_cb,checkpoint]

## compile model
M1.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['ac


## Trainning


M1.fit(X_train,Y_train,epochs=20,verbose=2,validation_data=(X_test,Y_test),batch_size =128,ca
```

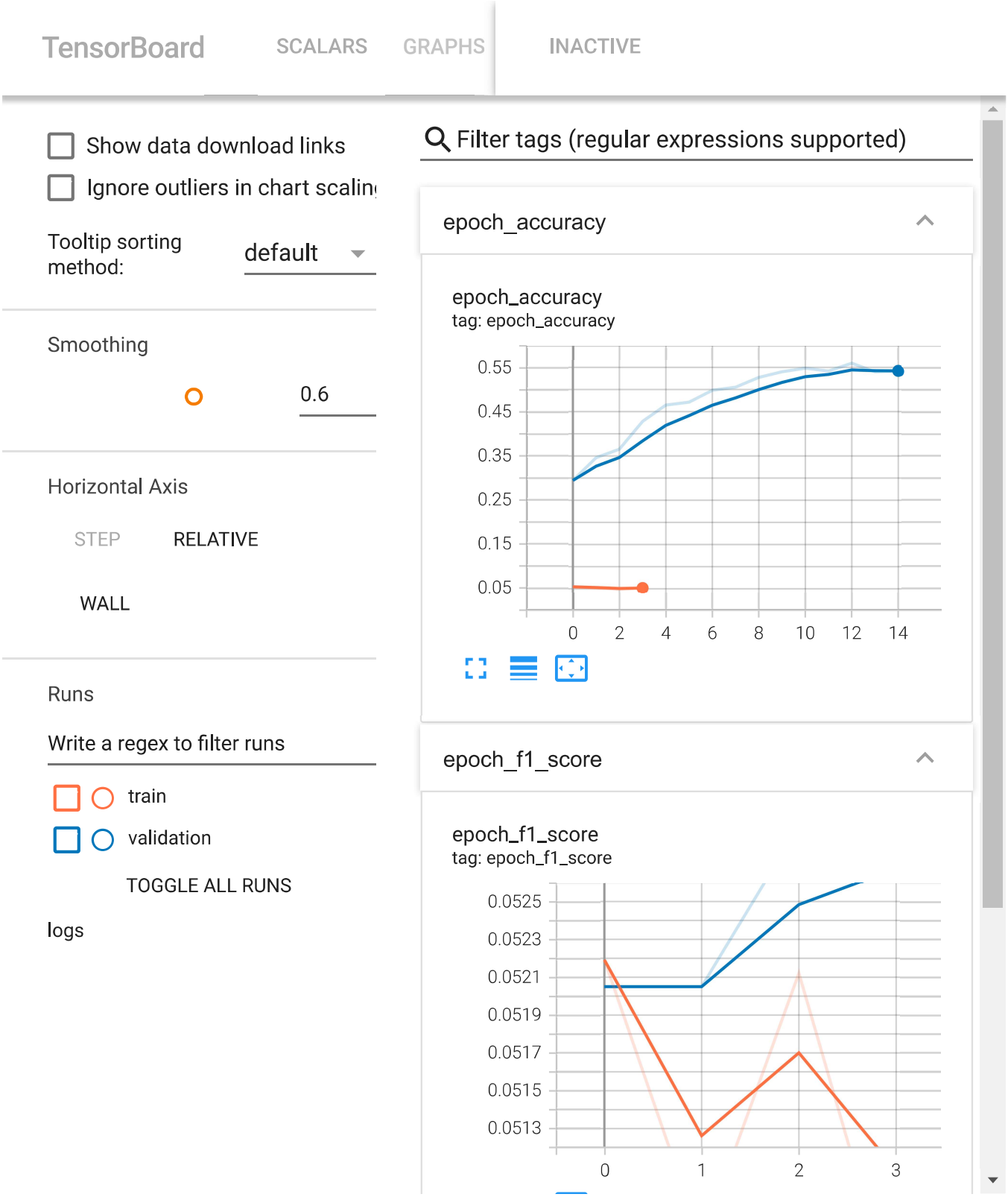
```
 Validation F1 score 0.36520076481835567

Epoch 00003: val_accuracy improved from 0.34608 to 0.36520, saving model to best_mode
Epoch 4/20
111/111 - 547s - loss: 2.2386 - accuracy: 0.4127 - val_loss: 2.2070 - val_accuracy: 0
 Validation F1 score 0.4280858296154663

Epoch 00004: val_accuracy improved from 0.36520 to 0.42809, saving model to best_mode
Epoch 5/20
111/111 - 546s - loss: 2.1249 - accuracy: 0.4441 - val_loss: 2.0940 - val_accuracy: 0
 Validation F1 score 0.46483960059485874

Epoch 00005: val_accuracy improved from 0.42809 to 0.46484, saving model to best_mode
Epoch 6/20
111/111 - 547s - loss: 2.0690 - accuracy: 0.4601 - val_loss: 2.0336 - val_accuracy: 0
 Validation F1 score 0.47142553643509666

Epoch 00006: val_accuracy improved from 0.46484 to 0.47143, saving model to best_mode
Epoch 7/20
111/111 - 544s - loss: 2.0210 - accuracy: 0.4784 - val_loss: 1.9575 - val_accuracy: 0
 Validation F1 score 0.4981941788825154

Epoch 00007: val_accuracy improved from 0.47143 to 0.49819, saving model to best_mode
Epoch 8/20
111/111 - 548s - loss: 1.9754 - accuracy: 0.4917 - val_loss: 1.9641 - val_accuracy: 0
 Validation F1 score 0.5052050138092203
```

```
Epoch 00008: val_accuracy improved from 0.49819 to 0.50521, saving model to best_mode
Epoch 9/20
111/111 - 547s - loss: 1.9373 - accuracy: 0.5063 - val_loss: 1.9129 - val_accuracy: 0
 Validation F1 score 0.5272997663055025

Epoch 00009: val_accuracy improved from 0.50521 to 0.52730, saving model to best_mode
Epoch 10/20
111/111 - 546s - loss: 1.9314 - accuracy: 0.5085 - val_loss: 1.8865 - val_accuracy: 0
 Validation F1 score 0.5400467388995114

Epoch 00010: val_accuracy improved from 0.52730 to 0.54005, saving model to best_mode
Epoch 11/20
111/111 - 545s - loss: 1.8793 - accuracy: 0.5271 - val_loss: 1.8821 - val_accuracy: 0
 Validation F1 score 0.5481198215423837

Epoch 00011: val_accuracy improved from 0.54005 to 0.54812, saving model to best_mode
Epoch 12/20
111/111 - 544s - loss: 1.8790 - accuracy: 0.5320 - val_loss: 1.8691 - val_accuracy: 0
 Validation F1 score 0.5417463352453792

Epoch 00012: val_accuracy did not improve from 0.54812
Epoch 13/20
111/111 - 546s - loss: 1.8437 - accuracy: 0.5422 - val_loss: 1.8618 - val_accuracy: 0
 Validation F1 score 0.5593796473337582

Epoch 00013: val_accuracy improved from 0.54812 to 0.55938, saving model to best_mode
Epoch 14/20
111/111 - 545s - loss: 1.8536 - accuracy: 0.5519 - val_loss: 1.8788 - val_accuracy: 0
 Validation F1 score 0.5398342893562779

Epoch 00014: val_accuracy did not improve from 0.55938
```

```
tf.keras.utils.plot_model(M1, to_file='M1.png', show_shapes=True, show_layer_names=True)
```




```
%tensorboard·--logdir·logs
```



Model2

```
#Splitting the data into train and test
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25,stratify=Y)
```

```
ke_tokenizer2= Tokenizer(filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n',char_level=True,oov_to

ke_tokenizer2.fit_on_texts(X_train)



print(ke_tokenizer2.word_index)      #Getting the word vocab of chars
```

```
{'UNK': 1, ' ': 2, 'e': 3, 't': 4, 'a': 5, 'o': 6, 'n': 7, 'i': 8, 's': 9, 'r': 10, 'h'
```

```
char_vocab_len=len(ke_tokenizer2.word_index)+1
char_vocab_len
```

```
41
```

```
#Performing Tokenization
X_train  = ke_tokenizer2.texts_to_sequences(X_train)
X_test = ke_tokenizer2.texts_to_sequences(X_test)


#Selecting sequencing length
#Checking 98th percenttile
per_tile_99=int(np.percentile(sents_len,99))
per_tile_99
```

```
8062
```

```
max_ch_len=per_tile_99
max_ch_len
```

```
8062
```

```
#Appliyng paddding based on maxlen
X_train=pad_sequences(X_train,max_ch_len,padding='post')
X_test=pad_sequences(X_test,max_ch_len,padding='post')


print(X_train.shape)
print(X_test.shape)
```

```
(14121, 8062)
(4707, 8062)
```

```
Y.shape
```

```
(18828, 20)
```

```
#Creating a character embedding matix
embd_ch_matrix=np.zeros((41,41))
for x,y in ke_tokenizer2.word_index.items():
      embd_ch_matrix[y][y]=1
```

```
embd_ch_matrix
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 0., 1.]])
```

```
embd_ch_matrix.shape
```

```
(41, 41)
```

```
#Model 2 Layers def
#Defining Embedding Layer
Embd_ly_char=Embedding(len(ke_tokenizer2.word_index)+1,41,embeddings_initializer=tf.keras.ini


input_layer2=Input(shape=(max_ch_len,))
embd_layer2=Embd_ly_char(input_layer2)
N1=Conv1D(64,3,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),kernel_
M1=Conv1D(64,3,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),kernel_

Max_pool1=MaxPool1D(5)(M1)
K1=Conv1D(64,3,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),kernel_
T1=Conv1D(64,3,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),kernel_
Max_pool2=MaxPool1D(4)(T1)
flat_layer1=Flatten()(Max_pool2)
dropout_layer1=Dropout(0.3)(flat_layer1)
dense_layer1=Dense(64,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(),
Output_layer1=Dense(20,activation='softmax',kernel_initializer=tf.keras.initializers.glorot_n
M2=Model(inputs=input_layer2,outputs=Output_layer1)
```

```
M2.summary()
```

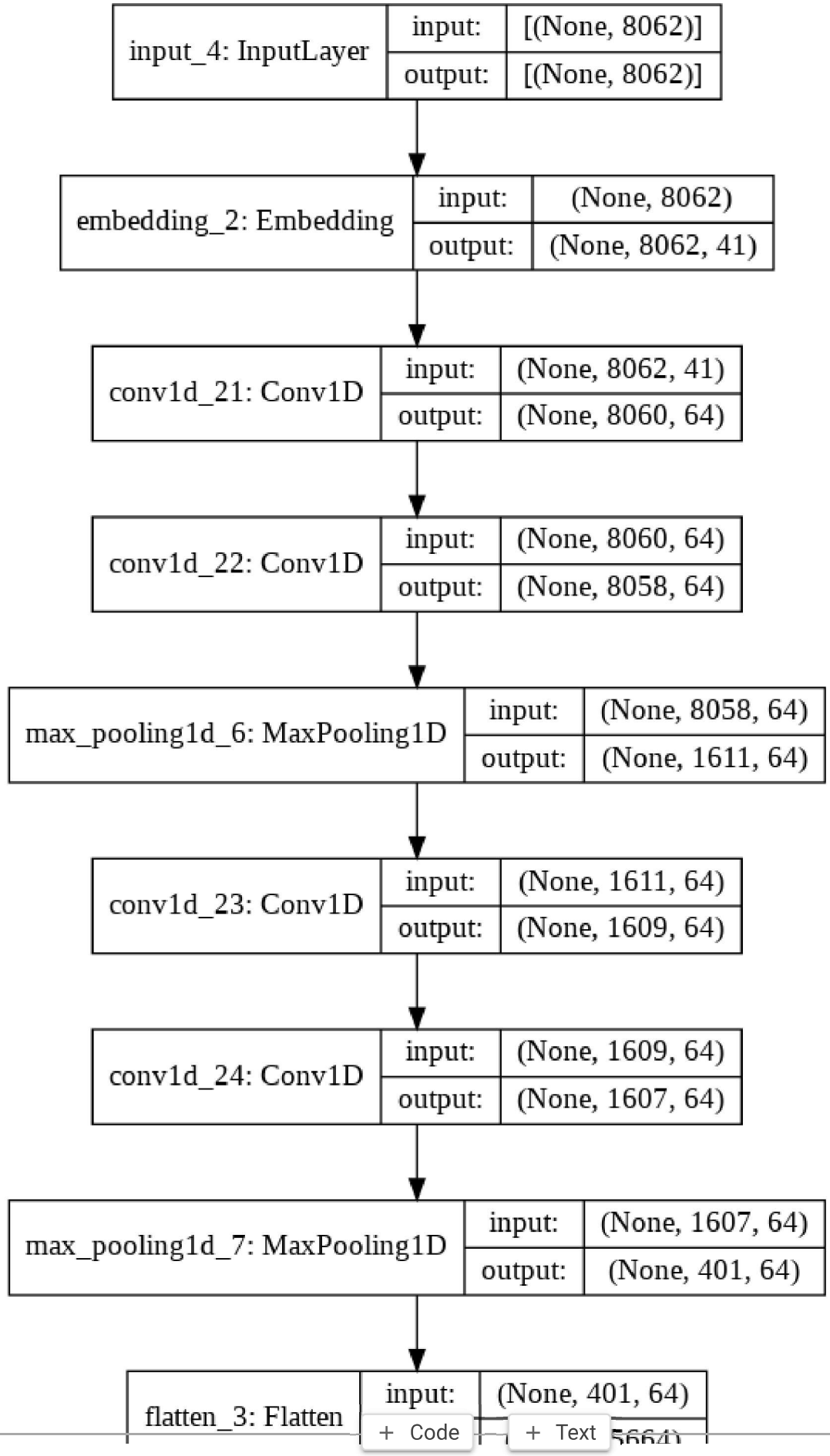
```
Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 8062)]            0
_________________________________________________________________
embedding_2 (Embedding)      (None, 8062, 41)          1681
_________________________________________________________________
conv1d_21 (Conv1D)           (None, 8060, 64)          7936
_________________________________________________________________
```

```
conv1d_22 (Conv1D)           (None, 8058, 64)          12352
_________________________________________________________________
max_pooling1d_6 (MaxPooling1 (None, 1611, 64)          0
_________________________________________________________________
conv1d_23 (Conv1D)           (None, 1609, 64)          12352
_________________________________________________________________
conv1d_24 (Conv1D)           (None, 1607, 64)          12352
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 401, 64)           0
_________________________________________________________________
flatten_3 (Flatten)          (None, 25664)             0
_________________________________________________________________
dropout_3 (Dropout)          (None, 25664)             0
_________________________________________________________________
dense_6 (Dense)              (None, 64)                1642560
_________________________________________________________________
dense_7 (Dense)              (None, 20)                1300
=================================================================
Total params: 1,690,533
Trainable params: 1,688,852
Non-trainable params: 1,681
_________________________________________________________________
```

```
## f1_score_callback
f1_metric=f1((X_test,Y_test))

## Callback for saving best model
checkpoint = ModelCheckpoint(filepath='best_model_1.h5',verbose=1,monitor='val_accuracy',
                             mode='max',save_best_only=True)

## Callback for earlystopping
early_stop = EarlyStopping(monitor="val_accuracy",mode='max',patience=2)

## Tensorboard
log_dir = "logs"
tensorboard_cb = TensorBoard(log_dir=log_dir,histogram_freq=1,write_graph=True)

#reduceon Plateau
reduce_cb=ReduceLROnPlateau(monitor='val_accuracy',factor=0.9,patience=1,verbose=1,mode='max'

## all callbacks
callbacks =[f1_metric,early_stop,tensorboard_cb,checkpoint,reduce_cb]

## compile model
M2.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['ac


## Trainning


M2.fit(X_train,Y_train,epochs=20,verbose=2,validation_data=(X_test,Y_test),batch_size =128,ca

     Epoch 1/20
```

```
111/111 - 240s - loss: 35.5907 - accuracy: 0.0554 - val_loss: 13.0939 - val_accuracy: 0
 Validation F1 score 0.052899936265137025

Epoch 00001: val_accuracy improved from -inf to 0.05290, saving model to best_model_1.h5
Epoch 2/20
111/111 - 241s - loss: 8.0364 - accuracy: 0.0513 - val_loss: 5.4160 - val_accuracy: 0.05
 Validation F1 score 0.052899936265137025

Epoch 00002: val_accuracy did not improve from 0.05290

Epoch 00002: ReduceLROnPlateau reducing learning rate to 0.0009000000427477062.
Epoch 3/20
111/111 - 239s - loss: 4.8527 - accuracy: 0.0514 - val_loss: 4.7086 - val_accuracy: 0.05
 Validation F1 score 0.053112385808370514

Epoch 00003: val_accuracy improved from 0.05290 to 0.05311, saving model to best_model_1
Epoch 4/20
111/111 - 249s - loss: 4.7344 - accuracy: 0.0530 - val_loss: 4.7371 - val_accuracy: 0.05
 Validation F1 score 0.053112385808370514

Epoch 00004: val_accuracy did not improve from 0.05311

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.0008100000384729356.
Epoch 5/20
111/111 - 247s - loss: 4.5389 - accuracy: 0.0530 - val_loss: 4.5227 - val_accuracy: 0.05
 Validation F1 score 0.053112385808370514

Epoch 00005: val_accuracy did not improve from 0.05311

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.0007290000503417104.
<keras.callbacks.History at 0x7fb6571bb1d0>
```

```
tf.keras.utils.plot_model(M2, to_file='M2.png', show_shapes=True, show_layer_names=True)
```



+ Code + Text

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 5878), started 0:38:49 ago. (Use '!kill 5878' to k
```

TensorBoard SCALARS GRAPHS INACTIVE

Show data download links

Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis

STEP RELATIVE

WALL

Runs

Write a regex to filter runs

train

validation

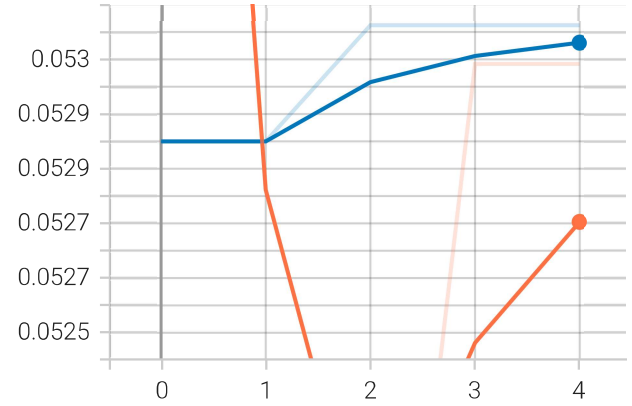
TOGGLE ALL RUNS

logs

Filter tags (regular expressions supported)

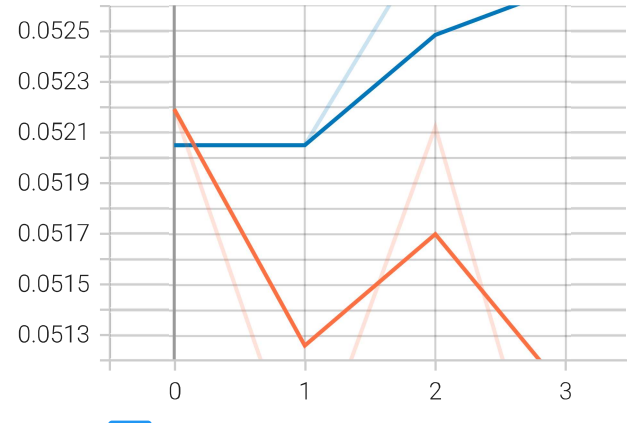
epoch_accuracy

epoch_accuracy
tag: epoch_accuracy



epoch_f1_score

epoch_f1_score
tag: epoch_f1_score



✓ 0s completed at 11:48 PM